Attor[redacted] Docket No.: 11333-014001

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: REPORTING THE STATE OF AN APPARATUS TO A REMOTE COMPUTER

APPLICANT: JAMES R. HANSEN

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL227256402US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

November 8, 2000
Date of Deposit

Samantha Bell
Signature

Samantha Bell
Typed or Printed Name of Person Signing Certificate

Attorney Docket No. 11333/014001

REPORTING THE STATE OF AN APPARATUS TO A REMOTE COMPUTER

Claim To Priority

This application is a continuation-in-part application of U.S. Patent Application No. 09/627,201, filed on July 28, 2000 in the name of James R. Hansen.

Background

This invention relates to using a device embedded in an apparatus (an "embedded device") to report the state of the apparatus to a remote computer.

An apparatus may contain an embedded device, such as a controller, to monitor and control its operation. Any type of apparatus may have an embedded device, including, but not limited to, home appliances, such as washing machines, dishwashers, and televisions, and manufacturing equipment, such as robotics, conveyors and motors.

Embedded devices are often connected to an internal network, such as a local area network (LAN), with an interface to the Internet. Other devices on the internal network may communicate with the embedded devices over the internal network.

Summary

In general, in one aspect, the invention is directed to using a device embedded in an apparatus to report the state of the apparatus to a remote computer. This aspect of the invention features detecting the state of the apparatus, generating a message that reports the state of the apparatus using a self-describing computer language, and sending the message to the remote computer. An example of a self-describing computer language is eXtensible Markup Language (XML). Examples of messages that may be sent include an electronic mail (e-mail) message and a hypertext transfer protocol (HTTP) command, both containing XML code.

By virtue of the device-generated message, the remote computer can obtain the state of the apparatus even if the remote computer cannot directly address the embedded device. Thus, computers that cannot communicate directly with the embedded device, such as computers that are not on the same internal network as the embedded device, can still obtain the status of the apparatus. Moreover, because the state is reported using a self-describing computer language, the remote computer can interpret the state without the aid of a person. As a result, processes, such

as maintenance and the like, can be scheduled automatically for the apparatus and/or embedded device by the remote computer.

This aspect of the invention may include one or more of the following features. The state is indicative of an error condition in the apparatus. The error condition is a variable that deviates from an acceptable value or a predetermined range of acceptable values. The function of detecting the state includes receiving the state from the apparatus by, e.g., retrieving the state periodically from the apparatus. The function of detecting the state includes obtaining an identifier for the apparatus, the identifier relating to the state of the apparatus, and using the embedded device to read the state from the apparatus using the identifier.

This aspect of the invention may also include determining if the state of the apparatus has changed. The message is generated if the state of the apparatus has changed and is not generated otherwise. The function of determining if the state of the apparatus has changed includes comparing the state received from the apparatus to a previous state of the apparatus.

The message is generated using a predefined template by obtaining one or more variables relating to the apparatus and inserting the one or more variables into the template. The state of the apparatus may be included as part of a body of an e-mail message or as part of an attachment to the e-mail message. The state of the apparatus may be included as part of an HTTP command.

In general, in another aspect, the invention is directed to obtaining a state of an apparatus from a device, such as a controller, embedded in the apparatus. This aspect of the invention features receiving a message that reports the state of the apparatus using a self-describing computer language and extracting the state of the apparatus from the message.

This aspect of the invention may include one or more of the following features. The self-describing computer language is XML. The state of the apparatus is indicative of an error condition in the apparatus. The error condition is a variable that deviates from an acceptable value or a predetermined range of acceptable values. The state of the apparatus is passed to a customer relationship management system. The message may be included in an HTTP

command or may be part of an e-mail.

In general, in another aspect, the invention features a system that includes first and second devices. The first device includes circuitry that generates a message reporting a state of an apparatus using a self-describing computer language. The second device is in communication with the first device. The second device includes circuitry that receives the electronic mail message from the first device.

This aspect of the invention may include one or more of the following features. The second device receives the message from the first device and extracts the state of the apparatus from the message. The first device is embedded in the apparatus and the second device is a remote computer. The message may be included in an HTTP command or may be part of an e-mail.

Other features and advantages of the invention will become apparent from the following description, including the claims and drawings.

Brief Description of the Drawings

Fig. 1 is a block diagram of a network containing a remote computer and an apparatus having an embedded device;

Fig 2 shows the format of a tag used to store state variables for the apparatus;

Fig. 3 is flowchart of a process performed by the embedded device to report the state of the apparatus to the remote computer;

Fig. 4 is a flowchart of an alternative process performed by the embedded device to report the state of the apparatus to the remote computer;

Fig. 5 is a flowchart of a process performed by the remote computer to interpret messages received from the embedded device; and

Fig. 6 is a block diagram of a network containing a remote computer and an apparatus having an embedded device the reports on the state of the apparatus using HTTP commands.

Description

Fig. 1 shows a network 10. Network 10 includes an apparatus 11 containing an embedded device 17, such as a controller (e.g., a microprocessor). Apparatus 11 is connected to an internal network 12, such as a LAN. A router or modem 14 interfaces internal network 12 to an external network 15, such as the Internet, that runs TCP/IP (Transmission Control Protocol/Internet Protocol) or some other suitable protocol. Connections may be, e.g., via Ethernet, wireless link, or telephone line. External network 15 contains remote computer 16, which may be a server, a personal computer (PC), or any other type of processing device. Other devices (not shown) may be included on internal network 12 and external network 15.

Processing In The Embedded Device

Apparatus 11 may be any type of device or may be included in any system having functions that are monitored and controlled by embedded device 17. Among other things, embedded device 17 executes software stored in memory 19 to generate and send, to remote computer 16, an e-mail message reporting the state of apparatus 11.

Software 20 includes an OPC (OLE for Process Control) server program 21, an XML (eXtensible Markup Language) processor program 24, and an e-mail program 25. E-mail program 25 is an SMTP-compliant (Simple Mail Transfer Protocol) program for sending e-mail from embedded device 17 to Internet addresses and for receiving e-mail from the Internet. E-mail program 25 operates as a mail transfer agent (MTA) for e-mail messages arriving at embedded device 17 and a mail delivery agent (MDA) for e-mail messages originating from embedded device 17. Other mail transfer protocols and programs may be also used by embedded device 17 in addition to, or instead of, those noted above.

XML processor program 24 is a program for generating XML code that reports the state of apparatus 11. XML is a self-describing computer language that defines variables and values relating to those variables. XML is self-describing in the sense that fields in the XML code identify variables and their values in the XML code. The template for XML used to generate an e-mail is as follows:

<name>temperature</name><value><##temperature##></value>,

where the "name" field identifies the name of a variable and the "value" field identifies the value of the variable that follows the "name" field. So, for the example given above, the variable is "temperature" and a value (e.g., 33.8) may be inserted for that variable as follows:

<name>temperature</name><value>33.8</value>.

XML processor program 24 generates XML code having the above syntax from a tag database 22 stored in memory 19.

Tag database 22 contains tags for use by XML processor program 24 in generating XML code. Fig 2 shows an example of a format for a tag 26, although other formats may be used. Tag 26 contains a name field 27, a description field 29, a value field 30, a time stamp field 31, and an item identifier (ID) field 32. These fields are used to obtain, identify and store information relating to apparatus 11.

Name field 27 holds the name of a state variable for apparatus 11, such as "temperature", and description field 29 provides further identification information, such as "temperature of fluid in a tank". Value field 30 holds the value of the state variable and time stamp field 31 holds

the time that the value in value field 30 was obtained. Value field 30 may include a variant, which is a construct that holds the value as an integer, a real number, a boolean, a character string, or some other type. Item ID field 32 holds an identifier that corresponds to hardware that is being monitored within apparatus 11. The identifier corresponds to a register location or to some other storage area of apparatus 11 that contains the value for field 30. For example, if embedded device 17 is in a robotics system, item ID field 32 might correspond to a register in the robotics system that contains a velocity or position of a robotic arm.

OPC server program 21 reads item IDs from field 32 and uses those item IDs to read variable values from corresponding hardware storage areas 34. OPC server program 21 implements an industrial automation protocol, such as MODBUS TCP, to communicate with the apparatus hardware. The system is not limited to use with the MODBUS protocol or with OPC server program 21; any drivers or computer programs may be used to read the state variable values from the hardware. Once a state variable value has been read, OPC server program 21 inserts the variable value

into field 30 of the appropriate tag.

Fig. 3 shows a process 36 for reporting the state of apparatus 11 to remote computer 16 using e-mail. In this embodiment, process 36 is implemented by OPC server program 21, XML processor program 24, e-mail program 25, and system software (not shown) executing in embedded device 17. The system software may include an operating system or other programs that control the background operation of embedded device 17.

Process 36 detects (301) the state of apparatus 11. The state may be indicative of an error condition (described below) within apparatus 11 or it may simply be state variables of apparatus 11 that are obtained at a particular time. To detect the state of apparatus 11, OPC server program 21 polls the hardware in apparatus 11 periodically. To perform this polling, OPC server program 21 obtains (301a) an item ID from tag database 22 and reads (301b) the value of a state variable that corresponds to the item ID from the appropriate hardware storage location. Process 36 may report the value to the remote computer as is or, alternatively, process 36 may use the value to identify and report an error condition in the hardware. A

process for reporting error conditions is described below.

Process 36 generates (302) an e-mail message reporting the value of state variable(s) for apparatus 11. Specifically, XML processor program 24 retrieves both the name of each state variable and the value of the state variable from the appropriate tag(s) in tag database 22. Other variables may also be retrieved from tag database 22 including the time stamp, description, and whatever other variables are stored in tag database 22. Which information is retrieved is pre-set in XML processor program 24. The retrieved variables are used by XML processor program 24 to generate XML code for an e-mail to remote computer 16.

XML processor program 24 may generate the XML code "on the fly", meaning without the use of a template. In this case, a blank XML file is populated with the retrieved variables in XML format by XML processor program 24. Alternatively, XML processor program 24 may generate the XML code using a pre-defined and formatted template. The template may be obtained by XML processor program 24, e.g., from memory 19 or a remote storage location (not shown). For example, the template may contain formatting similar to that shown above, namely:

<name>temperature</name><value><##temperature##></value>.

To generate the XML code from the template, XML processor program 24 scans through the template and inserts state variable value(s) retrieved from tag database 22, where appropriate. XML processor program 24 may generate the XML code periodically, depending upon how often e-mails are to be sent to the remote computer. Alternatively, tag manager software (not shown) may be included to provide newly-received tag variables to XML processor program 24. In this case, XML processor program 24 generates the XML code when it receives the new tag variables.

The resulting XML code may be part of the body of an e-mail or it may part of an attachment to an e-mail. The e-mail also contains a unique identifier, such as a code (e.g., serial number or identifier), that identifies embedded device 17 to remote computer 16. E-mail program 25 obtains the XML code from XML processor program 24 and sends it to remote computer 16 as part of the e-mail message. E-mail program 25 obtains the code periodically, depending upon the frequency at which e-mails are to be

sent to the remote computer. The frequency is set beforehand in embedded device 17. The address of the remote computer may be registered with e-mail program 25 beforehand. Typically, the address/remote computer will be that of an entity that requires information about apparatus 11. For example, the entity may be a manufacturer of the apparatus, a plant monitoring system, or the like. The e-mail program sends the message to router/modem 14, which transfers it via external network 15 to remote computer 16. Then, the e-mail message is processed as described below.

The foregoing describes the case where embedded device 17 simply reports the state of apparatus 11 to remote computer 16 periodically. Alternatively, embedded device 17 may report the state to remote computer 16 only when an error condition or "alarm" is detected.

Fig. 4 shows a process 40 by which embedded device 17 detects error conditions in apparatus 11 and sends an e-mail message to remote computer 16 when an error condition is detected. Process 40 detects (401) the state of apparatus 11, where, as above, "state" refers to tag variable values for apparatus 11. Detection (401) is performed in the same manner as process 36; therefore, a

description is omitted here. Once process 36 has obtained the state of apparatus 11, process 36 determines (402) if that state represents an error condition.

To detect an error condition, process 40 may compare an obtained state variable value to a predetermined acceptable value or a range of predetermined acceptable values. If the state variable value is outside the range of, or deviates considerably from, the acceptable value(s), then process 40 knows that an error condition is present. Alternatively, process 40 may store each state variable value in memory 19 as it is obtained, and compare each newly-received state variable value to one or more stored state variable values. If the new state variable value deviates by more than a predetermined amount from the stored value(s), process 40 knows that an error condition is present/has occurred.

An error condition may be based on a single state variable value or it may be based on some combination of two or more state variable values. For example, if embedded device 17 is in manufacturing equipment that monitors both a level of fluid in a tank and a temperature of that fluid, an error condition may only be present if

both the fluid level and the temperature exceed preset values. In this example, therefore, if only one state variable exceeds its corresponding preset value, then no error condition is present/has occurred.

If process 40 detects (402) an error condition, process 40 generates (403) an e-mail message and sends (404) the e-mail message to remote computer 16. The functions of generating and sending an e-mail message are performed as described above with respect to process 36; therefore, detailed descriptions are omitted here. When generating the e-mail message, e-mail program 25 may place the state variable(s) that caused the error condition in the "subject" line of the e-mail. If process 40 does not detect (402) an error condition, an e-mail message is not sent, whereafter process 40 returns to 401.

XML processor program 24 may maintain a log of error conditions in memory 19. This error condition "history" may be provided along with each new e-mail message. The history may relate to a particular state variable or to more than one state variable. For example, if the error condition pertains to temperature, XML processor program 24 may include the error condition history for temperature in

the e-mail. If the error condition pertains to both temperature and tank level, XML processor program 24 may include the error condition history for both temperature and tank level in the e-mail. If a template is used to generate the e-mail message, portion(s) of that template may be reserved for error condition history.

Processes 36 and 40 can be combined to generate an e-mail periodically that reports the state of apparatus 11 to remote computer 16 even if no error conditions have been detected in apparatus 11, and that also flags any error conditions if any have been detected. XML processor program 24 adds an indicator or the like next to state variable values that correspond to error conditions.

Processes 36 and 40 may be executed by embedded device 17 to monitor and report on any type of state variables in any type of apparatus. For example, processes 36 and 40 may detect state variable values relating to conveyor belt speed, current and/or voltage in electronic devices, tank fluid levels, input/output sensors, and the like. Processes 36 and 40 may detect state variable values through a programmable logic controller (PLC) that is connected to one or more other devices. A PLC includes

plug-in cards for each device that obtain and store device state variable values. OPC server program 21 communicates with these plug-in cards to obtain the device state variable values for generating e-mails as described above.

E-mails generated by processes 36 and 40 report the state of apparatus 11 using a self-describing computer language, such as XML; however, other types of self-describing computer languages may be used. In addition, other text and/or images may be included in the e-mails, if desired and appropriate under the circumstances. Described below is a process that is performed by remote computer 16 to interpret e-mails received from embedded device 17.

Processing In The Remote Computer

Remote computer 16 contains a controller 41 for executing software stored in memory 42. Among this software is e-mail program 44, XML parser 45, and customer relationship management (CRM) system software 46.

As in embedded device 17, e-mail program 44 is an SMTP-compliant program for receiving e-mail from embedded device 17 and other such devices. E-mail program 44 operates as a mail transfer agent (MTA) for e-mail messages

arriving at remote computer 16 and a mail delivery agent (MDA) for e-mail messages originating from remote computer 16. E-mail program 44 uses the same protocol as e-mail program 25 in embedded device 17.

XML parser 45 parses XML code in a received e-mail to extract variable values, including an identifier for apparatus 11. XML parser 45 recognizes field names, such as "name" and "value" from above and extracts corresponding state variable values from those fields. That is, XML parser 45 knows the syntax of XML. Knowing this, XML parser 45 is able to extract variable names from the "name" fields, corresponding variable values from the "value" fields, and any other information in the XML code.

XML parser 45 passes the state variable values, along with appropriate identifiers, to customer relationship management system software 46 or whatever other software or database requires/uses those state variable values.

Fig. 5 shows how an e-mail from embedded device 17 is processed (43). Once an e-mail has been received (501) from embedded device 17, XML parser 45 extracts (502) the state variable values of apparatus 11 from the e-mail. For example, XML parser 45 may extract tank levels, temperature

values, etc., of apparatus 11 monitored by embedded device 17. The state variable values may be indicative of error conditions in apparatus 11, as defined above, or simply state variables for apparatus 11 obtained at a given point in time.

XML parser 45 passes (503) the state variable values, i.e., the state of apparatus 11, to customer relationship management system software 46. Customer relationship management system software 46 uses these state variable values, e.g., to schedule maintenance for apparatus 11 if necessary, to provide software upgrades to apparatus 11, or for any other purpose. Because the XML code in the e-mail is readable by XML parser 45, reporting and scheduling by customer relationship management system software 46 can be done automatically. It is noted that e-mail program 44 may still forward an e-mail to a customer representative, technician, or the like, particularly if an e-mail contains human-readable text.

The software on remote computer 16 is not limited to that shown in Fig. 1. For example, XML parser 45 may be replaced by a parser that is capable of parsing/reading other types of computer code, depending upon the code that

is used in the received e-mail. Likewise, the parsed variables can be passed to software other than customer relationship management system software 46. For example, the variables can be stored in a database 47 for later use.

Alternative Embodiment

Referring to Fig. 6, a network 60 is shown on which an alternative embodiment of the invention is implemented. Network 60 is identical to network 10, except that e-mail program 25 in apparatus 11 is replaced by Web client 61 and e-mail program 44 in remote computer 16 is replaced by Web server 62. This alternative configuration allows embedded device 17 to transfer messages to remote computer 16 as HTTP commands rather than e-mails.

The HTTP command may be an HTTP POST command, although other HTTP commands, such as an HTTP GET command, may instead be used. An example of an HTTP POST command that uses XML code to report the status of a fictitious "widget" apparatus is as follows:

```
POST /CONTROL HTTP/1.1
Host: www.acme.com
Content-Type: text/xml
Content-length: nnn

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:www-acme-
com:device:Widget:3</deviceType>
    <friendlyName>Widget</friendlyName>
    <manufacturer>Acme Industries</manufacturer>
    <modelName>Widget</modelName>
    <modelNumber>3</modelNumber>
    <serialNumber>53266D</serialNumber>
    <UDN>uuid:4A89EA70-73B4-11d4-80DF-0050DAB7BAC5</UDN>
  </device>
</root>
<parameters>
  <Airflow xsd:type="integer">378</Airflow>
  <Humidity xsd:type="double">46.7</Humidity>
  <Motor xsd:type="integer">1500</Motor>
  <Vent xsd:type="integer">4</Vent>
</parameters>
<alarms>
  <Temperature>
    <description>Room temperature is above
83F</description>
    <severity>300</severity>
    <status>high</status>
  </Temperature>
</alarms>
```

XML is a self-describing computer language in the sense that fields in the XML code identify variables and their values in the XML code. For example, as shown in the

above POST command, the "manufacturer" field identifies a manufacturer, e.g., "Acme Industries", and is delineated by "<manufacturer>" to indicate the start of the field and "</manufacturer>" to indicate the end of the field. XML is used in the HTTP command because it can be generated, parsed and read relatively easily by XML parser 45.

The HTTP POST command includes data identifying apparatus 11. This data includes, but is not limited to, data identifying the type of the device, a common (or "friendly") name for the device, the manufacturer of the device, the model name of the device, the model number of the device, the serial number of the device, and a universal unique identifier (UUID) for the device. In the example post command, this data is formatted as:

```
<friendlyName>Widget</friendlyName>
<manufacturer>Acme Industries</manufacturer>
<modelName>Widget</modelName>
<modelNumber>3</modelNumber>
<serialNumber>53266D</serialNumber>
<UDN>uuid:4A89EA70-73B4-11d4-80DF-0050DAB7BAC5</UDN>
```

The HTTP POST command also provides the state of apparatus 11. The state includes operational parameters and alarm conditions for apparatus 11. In the above HTTP POST command, these are formatted as follows:

```
<parameters>
  <Airflow xsd:type="integer">378</Airflow>
  <Humidity xsd:type="double">46.7</Humidity>
  <Motor xsd:type="integer">1500</Motor>
  <Vent xsd:type="integer">4</Vent>
</parameters>
<alarms>
  <Temperature>
    <description>Room temperature is above
83F</description>
    <severity>300</severity>
    <status>high</status>
  </Temperature>
</alarms>
```

Thus, the state of the widget includes information on its airflow, humidity, motor and vent settings, temperature, severity of the temperature, and temperature status. Different information from that shown may be included in the HTTP POST command.

Referring back to Figs. 3, 4 and 5, in this embodiment the operation of processes 36, 40 and 43 is identical to that described above, except that, in all steps, the e-mail message is replaced by an HTTP command. In apparatus 11, the HTTP command is generated by Web client 61 based on data provided by XML processor 24. This XML data is the same as that used above with e-mail program 25. Embedded device 17 sends the HTTP command to remote computer 16, where it is received by Web server 62 and then processed by

XML parser 45. Thereafter, processing proceeds as above.

Architecture

Processes 36, 40 and 43 are not limited to use with the hardware/software configuration of Fig. 1; they may find applicability in any computing or processing environment. Processes 36, 40 and 43 may be implemented in hardware (e.g., an ASIC {Application-Specific Integrated Circuit} and/or an FPGA {Field Programmable Gate Array}), software, or a combination of hardware and software.

Processes 36, 40 and 43 may be implemented using one or more computer programs executing on programmable computers that each includes a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. Also, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on a storage medium or device (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform processes 36, 40 and 43.

Processes 36, 40 and 43 may also be implemented as a computer-readable storage medium, configured with a computer program, where, upon execution, instructions in the computer program cause the computer to operate in accordance with processes 36, 40 and 43.

Other embodiments not described herein are also within the scope of the following claims. For example, e-mail or http messages sent from apparatus 11 to remote computer 16 may be queued (e.g., stored in memory 19) and then retrieved and sent out at a later time. Queuing messages reduces message loss resulting from intermittent system failures.

What is claimed is: